Abstract

Social media has in increasing impact on society in our current society. Information is constantly being shared and consumed influencing opinions, educating and in some cases sharing destructive ways of thinking. A major component in this are social media influencers. These influencers have a lot of followers and information is easily passed onto them to see, and likely for them to repost for others to see. Influencers hold much responsibility as the flow of information to their followers is a very strong connection.

Background

The simulation code continues to step through time across each of the posts in the network until the user tells it so stop. It saves the current network state, such as users and the different follows between users and then also prints the current statistics for probability of like and follows. It also saves each of the posts showing the actual post text itself, who it was posted by, how many people have seen it, the people that have liked the post and also the time since the post has been posted.

The simulation code reads from two files, a network file with all the users and follows in the network state. But then goes one by one through an additional events file where a user could be added, a follow occur or a post be created. The network file is read first, and the event file second. In my version of the simulation code I decided to write the network state before the timestep so we can see the original state of the network at the very second the post is created.

The simulation code uses an integer of time where we say time is just a propagation rather than actual seconds and hours. This just makes it easier to see each level of the graph be traverses for each post. The investigation is into the spread of information because of the original poster’s attributes and thus actual time is not a factor.

The information we are mostly looking at is the number of people that have seen the post. This is displayed as seenBy which is a length method on a linked list of all the users that have seen the post. This value will then be compared with the time since the post was made to see the impact of followers of the influencer has on the flow of information.

Methodology

For the simulation code, the time complexity of the all the fileIO methods – write, process and read – have a time complexity of O(N) as we only traverse through the amount of lines there are in the file or the lines needed to write to the file. Memory wise this is something we cannot improve as its dependent on the buffer and how the operating system and coding language deal with fileIO.

The timestep method includes a traversal of 2 or 3 linked lists. First we traverse through every post in the network. If the post wasn't just created – ie. Have time=0 – then we traverse through the linked list of the current users to show the post to. If it was just created we skip this part as there are no previous people to show the post to because the original poster (influencer) has already seen the post as they posted it. Then in both situations the adjacency list of the people either following the poster(influence) or the people who have liked the post. Additionally with this timestep we do this as many times as the time+1 is. So if time is equal to 0 then we only do the traversals once, compared to twice is time is equal to 2. The time complexity for this then would be around O(T(P+S/2+F) where time is equal to T, number of posts is P, number of people to show if S and number of followers that person has is F. With memory there is a bit of a waste of memory as we have 3 linked lists with the show post list getting replaced with the check list every time anyway.

Adding a user has a very low time complexity of O(N). The actual insert of a user is O(1) however, we check first that the user does not already exist, which requires a traversal of the linked list of vertices in the graph before simply inserting last into the vertex list. We use a double-linked, double-ended linked list for the graph, however for the add user function we only need a double-ended to add to the end. So the double-linked previous node storage isn't perfect for memory performance.

The add follow method has a time complexity of O(N) as it is a directional graph. The program traverses the linked list of users until the follower is found and then adds the user that will follow the follower to the adjacency list. Memory wise we are again using a double-linked, double-ended linked list despite only using insert last and thus a double-linked list could be said to be a waste of memory.

The add post method is simply O(1) as we are adding a post object to the end of the posts linked list within the network. Once more we can say that there is a waste of memory as we only use insert last with this linked list and not utilise the double-linked memory stored.

The actual building of the string to output to the file includes the display posts methods which includes a traversal of the posts linked list thus O(N). The outputFIle method is also called which traverses through the users and then also traverses through the people following that each user which is O(U + F) where U is number of users and F is number of follow relationships. Thus the method has a time complexity of O(N + U + F). *Note N is number of posts.*

Overall the simulation does not delete data and thus a double-linked linked list is not needed as we never have to do a removeLast.

For the simulation, I have chosen to compare multiple runs of the program by the number of people who have seen the post at each time stop. This is because I want to measure the number of people who see the post over time due to the effects of the number a follower the poster has.

The 0.7 is a 70% chance they will like the post which is a reasonable assumption as people are following users for their content and thus likely to like the content. There is a 40% chance of following if they haven't which in an influence sense may be somewhat likely to happen but not to a high degree. These probabilities are kept the same in all trial runs of the simulation as we are not testing this change.

For the first run of the simulation trial we try with 9 people following the poster(influencer) and there is a total of 20 people following them with them randomly following the 9 people. The files used for this run of the simulation is test1NW (NW is network file) and also test1EF (EF is event file). This trial run of the simulation is run using "java SocialSim -s test1NW test1EF 0.7 0.4".

For the second run of the simulation we try 4 people following the poster(incluencer) with a total of 5 people following those 4 (ie. The rest of the nine from the first test) and then 20 people randomly following those 9 followers. The files used for this run of the simulation is test2NW (NW is network file) and also test1EF (EF is event file). This trial run of the simulation is run using "java SocialSim -s test2NW test1EF 0.7 0.4".

For the third run of the simulation we try 1 people following the poster(incluencer) with a total of 8 people following those 1 people (ie. The rest of the nine from the first test) and then 20 people randomly following those 9 followers. The files used for this run of the simulation is test3NW (NW is network file) and also test1EF (EF is event file). This trial run of the simulation is run using "java SocialSim -s test3NW test1EF 0.7 0.4".

For the fourth run of the simulation we try 0 people following the poster(incluencer) with a total of 9 people following no one and then 20 people randomly following those 9 followers. The files used for this run of the simulation is test4NW (NW is network file) and also test1EF (EF is event file). This trial run of the simulation is run using "java SocialSim -s test4NW test1EF 0.7 0.4".

The first test contains data represents a user with 9 followers, test2 a user with 4 followers, test3 a user with 1 follower and a base case of test4 with no follower. The purpose of this is to see the effects on the original poster count on the traversal of data without changing the number of users in the system, only changing the number of follows to the poster(influencer).

Results

Note that in the tables the trial columns represent the amount of users that saw the post per trial run. Also note that due to the test data the end of the graph is reached at time step 2 in test1 and at step3 in test2. This is due to the arrangement of follows

For the first test which has 9 follower following the poster:

Time	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Avg No.
0	0	0	0	0	0	0
1	9	9	9	9	9	9
2	11	10	10	9	10	10
3	11	10	10	9	10	10
4	11	10	10	9	10	10

For the second test which has 4 follower following the poster:

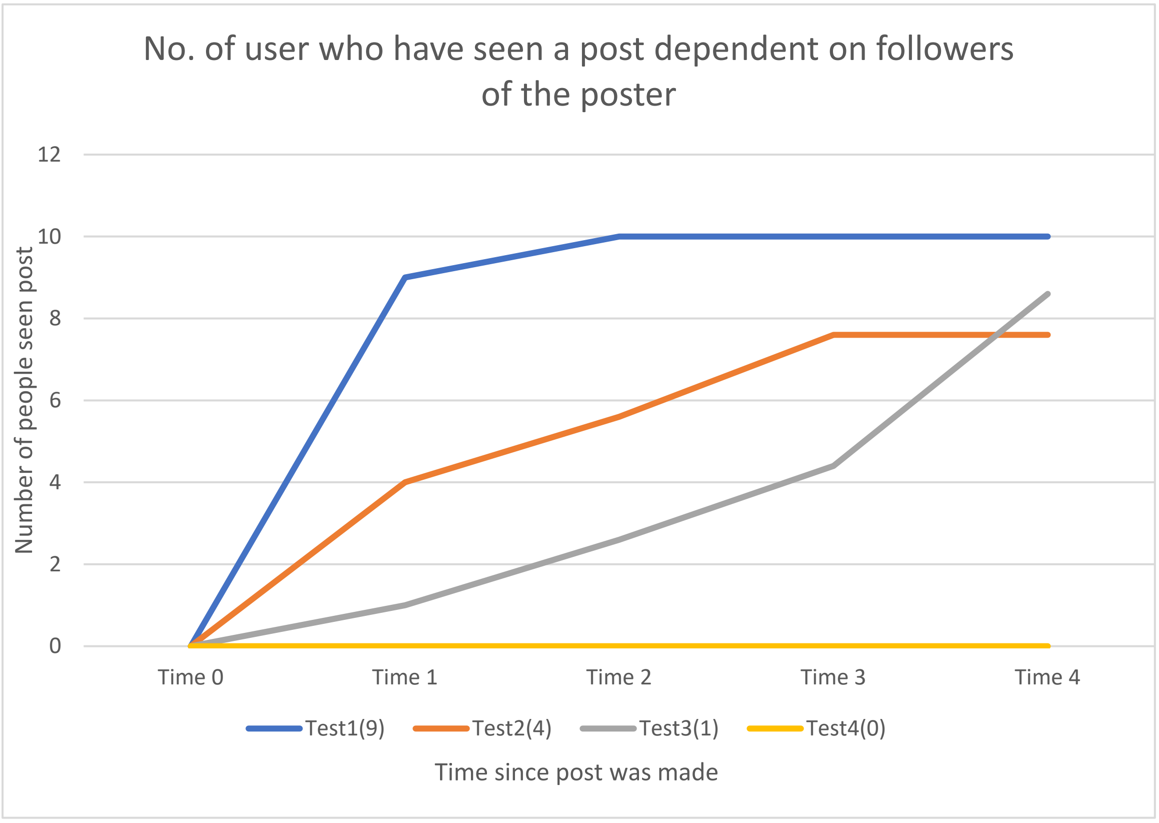
Time	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Avg No.
0	0	0	0	0	0	0
1	4	4	4	4	4	4
2	7	5	7	4	5	5.6
3	9	6	11	4	8	7.6
4	9	6	11	4	8	7.6

For the second test which has 1 follower following the poster:

Time	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Avg No.
0	0	0	0	0	0	0
1	1	1	1	1	1	1
2	4	4	3	1	1	2.6
3	10	6	4	1	1	4.4
4	21	10	8	3	1	8.6

For the second test which has 0 follower following the poster:

Time	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Avg No.
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0



As part of the results we can see that the test 1 consistently resulted in higher people seeing the post. This would have been due to it not being left up to probability whether the followers will like and then let other people see the post which is why the line flattens off rather quick after time 1.

Test 2 came out with some average results, staying somewhere between test 1 and test3 as it was a median test to see a slight variation on the number of people following.

Test 3 has some varied results with some being incredibly small amount of people seeing the post and also even having a case where 21 people saw the post which was the maximum number of people watching the post in any category. This was due to there being only one follower to the influencer and was more dependent on the probabilities to see how many people would like and then see the post. We also saw a case with a lot of ones which was due to the fact that it wasn’t liked by the follower of the poster and therefore only that follower saw the post as it wasn’t showed to their followers.

Test 4 was a base case in which no one could see the post as no one is following the original poster.

Looking at the graph we can distinguish two main features of the lines and how they compare to each other. Test1 has a very steep gradient from time0 to time compared to that of Test3 and Test4. This is because all of the followers of the poster will see the post no matter what in Test1 so 9 people will always be shown compared to 1 for test3.

We also notice the way the line steepness changes over time. Test1 has a steep gradient at the beginning and then flattens off whilst tes3 almost has an exponential or linear shape to it. This is because test3 continue to slowly show its posts to people over time but slower than test1 which shows a lot of people at first and slowly plateaus out.

Conclusion and future work

The conclusions shown by the results and the graph shows that there is a big difference in the way information propagates through social networks when dealing with the original followers of social media influencers. The followers will always be shown the information an it quickly spreads out to the surrounding network. It showcases the power that these people can have over others when sharing information.

The results match my prediction where a big impact of information propagation is due to the original followers.

Some further investigations could go into the age of users and how that may affect data propagation. Whether younger users are more likely to share something when compared to an older. Another could be gender, where perhaps a certain gender may be more likely to share a post and thus propagate it thorough the network.

The liking system of the network is just a probability, while in real life it is dependent on the nature of the post and the context of the user. Some more investigation into user contexts and the nature of posts affects the propagation of the information through the network.

Further investigation into the effects of number of posts could be undertaken as well, as more posts would mean more information through the list and also more propagations and chanced of liking and following and post and thus resulting in more people seeing the post.